Midterm 2 - Section 002 (C)

Name:

Penn State access ID (xyz1234) in the following box:

Student ID number (9XXXXXXXX):

Instructions:

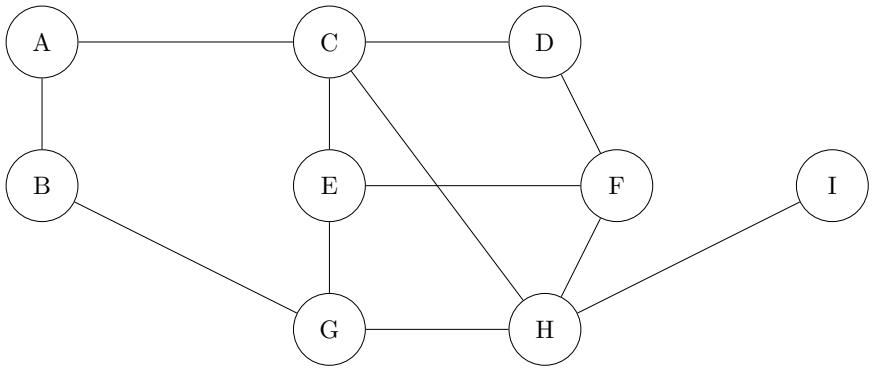
- Answer all questions. Read them carefully first. Be precise and concise. Handwriting needs to be neat. Box numerical final answers.
- Please clearly write your name and your PSU access ID (i.e., xyz1234) in the box on top of **every page**.
- Write in only the space provided. You may use the back of the pages only as scratch paper. **Do not write your solutions on the back of pages!**
- **Do not write outside of the black bounding box**: the scanner will not be able to read anything outside of this box.
- We are providing one extra page at the end if you need extra space. Make sure you mention in the space provided that you answer continues there.

Good luck!

Name:	**PSU Access ID (xyz1234):**

Graphs *(28 points)*

1. Perform depth-first search (DFS) on the following graph; whenever there is a choice of vertices, pick the one that is alphabetically first. Find the pre and post visit numbers for each vertex.



Pre-visit numbers:

(a) The pre-visit number of node A is:

●1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(b) The pre-visit number of node B is:

◯1 ●2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(c) The pre-visit number of node C is:

◯1 ◯2 ◯3 ◯4 ●5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(d) The pre-visit number of node D is:

◯1 ◯2 ◯3 ◯4 ◯5 ●6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(e) The pre-visit number of node E is:

◯1 ◯2 ◯3 ●4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(f) The pre-visit number of node F is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ●7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(g) The pre-visit number of node G is:

◯1 ◯2 ●3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(h) The pre-visit number of node H is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ●8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(i) The pre-visit number of node I is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ●9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

Name:	**PSU Access ID (xyz1234):**

Post-visit numbers:

(j) The post-visit number of node A is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ●18

(k) The post-visit number of node B is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ●17 ◯18

(l) The post-visit number of node C is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ●14 ◯15 ◯16 ◯17 ◯18

(m) The post-visit number of node D is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ●13 ◯14 ◯15 ◯16 ◯17 ◯18

(n) The post-visit number of node E is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ●15 ◯16 ◯17 ◯18

(o) The post-visit number of node F is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ●12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(p) The post-visit number of node G is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ●16 ◯17 ◯18

(q) The post-visit number of node H is:

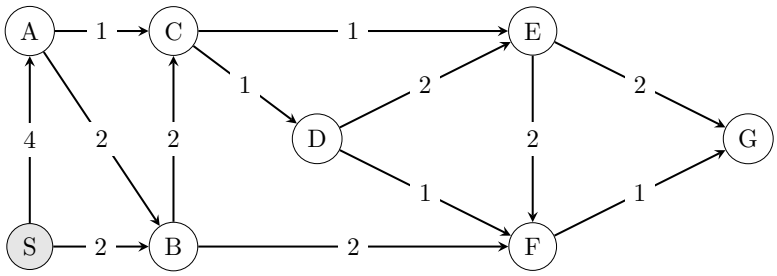
◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ●11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

(r) The post-visit number of node I is:

◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ●10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

Name:	**PSU Access ID (xyz1234):**

2. Run Djikstra's Priority Queue Algorithm for three iterations starting with the source node S, and provide the shortest path distances obtained in the below graph.



(a) Distance to S at end of 3rd iteration:

●0 ◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ◯∞

(b) Distance to A at end of 3rd iteration:

◯0 ◯1 ◯2 ◯3 ●4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ◯∞

(c) Distance to B at end of 3rd iteration:

◯0 ◯1 ●2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ◯∞

(d) Distance to C at end of 3rd iteration:

◯0 ◯1 ◯2 ◯3 ●4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ◯∞

(e) Distance to D at end of 3rd iteration:

◯0 ◯1 ◯2 ◯3 ●4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ◯∞

(f) Distance to E at end of 3rd iteration:

◯0 ◯1 ◯2 ◯3 ◯4 ●5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ●∞

(g) Distance to F at end of 3rd iteration:

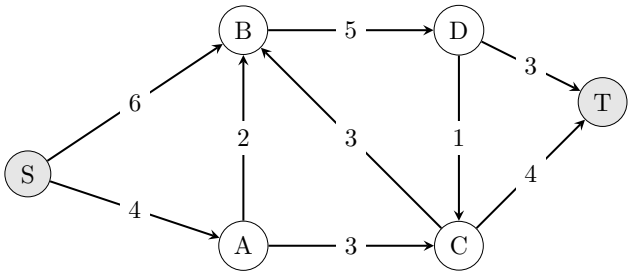
◯0 ◯1 ◯2 ◯3 ●4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ●∞

(h) Distance to G at end of 3rd iteration:

◯0 ◯1 ◯2 ◯3 ◯4 ●5 ◯6 ◯7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18 ●∞

Name:	**PSU Access ID (xyz1234):**

3. Given the flow network depicted below with source node S and sink node T, fill in the values for the residual capacities after Ford-Fulkerson terminates (both forward and backward even if it is zero). What is the maximum amount of flow that the network would allow to flow from source to sink node? Note that how you find, and in what order, (S, T)-augmenting paths does not matter.



This question has too many possible answers. One of those answers is included, however, points are awarded to other answers as well.

(a) Edge (S, B):
●0 ◯1 ●2 ●3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(b) Edge (B, S):
◯0 ◯1 ◯2 ●3 ●4 ◯5 ●6 ◯7 ◯8 ◯9 ◯10

(c) Edge (S, A):
●0 ●1 ◯2 ●3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(d) Edge (A, S):
◯0 ●1 ◯2 ●3 ●4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(e) Edge (A, B):
●0 ●1 ●2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(f) Edge (B, A):
●0 ●1 ●2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(g) Edge (T, B):
●0 ◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(h) Edge (B, T):
●0 ◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(i) Edge (A, C):
●0 ●1 ●2 ●3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(j) Edge (C, A):
●0 ●1 ●2 ●3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(k) Edge (C, T):
●0 ◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(l) Edge (T, C):
◯0 ◯1 ◯2 ◯3 ●4 ◯5 ◯6 ◯7 ◯8 ◯9 ◯10

(m) The value of the maximum flow is:
◯0 ◯1 ◯2 ◯3 ◯4 ◯5 ◯6 ●7 ◯8 ◯9 ◯10 ◯11 ◯12 ◯13 ◯14 ◯15 ◯16 ◯17 ◯18

Name:	PSU Access ID (xyz1234):

True/False *(20 points)*

True or false? Fill in the correct bubble. No justification is needed.

	T	F	
1.	●	○	Given a valid pre-order numbering of vertices of a directed graph, there is only one corresponding post order numbering of vertices.
2.	○	●	Let P be a shortest path from some vertex s to some other vertex t in a directed graph. If the weight of each edge in the graph is increased by one, P will still be a shortest path from s to t.
3.	●	○	If a topological sort exists for the vertices in a directed graph, then a DFS on the graph will produce no back edges
4.	●	○	Suppose we do a DFS on a directed graph G. If we remove all of the back edges found, the resulting graph is now acyclic.
5.	○	●	For all weighted graphs and all vertices s and t, Bellman-Ford starting at s will always return a shortest path to t.
6.	○	●	For a directed graph, the absence of back edges with respect to a BFS tree implies that the graph is acyclic.
7.	●	○	In an unweighted graph where the distance between any two vertices is at most D, any BFS tree has depth at most D, but a DFS tree might have larger depth.
8.	○	●	After running depth-first search on a directed graph, the node with the smallest post number is part of a source component.
9.	●	○	If vertices u, v are in the same strongly connected component of a directed graph G, then it is necessarily the case that v is reachable from u in G.
10.	○	●	The shortest path between two vertices is unique if all edge weights are distinct.

Name:	PSU Access ID:

Short Questions. *(20 points)*

1. We are given a directed graph G with positive weights on its edges. We wish to find a shortest path from s to t, and, among all shortest paths, we want the one in which the longest edge (i.e. edge with largest weight) is as short as possible. How would you efficiently modify Dijkstra's algorithm to this end?

   Solution: Use a tuple (i, w_i) to indicate the value of the longest edge along the path to node i. Each time when we update the shortest distance to a node i, we will also update w_i. Let the previous node of i be j. Then we set w_i to $\max\{w_j, C(j,i)\}$. If the distance to node i is the same via nodes j and k, we will pick the node with smaller longest edge, i.e. $\min\{\max\{w_j, C(j,i)\}, max\{w_k, C(k,i)\}\}$. Since we just add one more step (i.e., updating longest edge value and picking smaller one) during the distance update procedure, the running time should be the same as Dijkstra's algorithm, which is $O((|V| + |E|) \log |V|)$.

2. Given an undirected graph with nonnegative edge costs, along with non-intersecting subsets S and T of vertices (i.e., $S \cup U = V$ and $S \cap U = \emptyset$), give a short description of an efficient method to determine the distance between the closest pair of nodes between S and T; that is, $u \in S, v \in T$ such that $\text{dist}(u, v)$ is minimal in terms of shortest path costs.

   Solution: Since all edges are nonnegative, the closest pair must be the edges across S and T. Iterate all edges to find the edges across subsets S and T. Then sort once to find the edge with minimum weight. The running time of both iteration and sorting are $O(E)$. Overall, the running time is $O(E)$.

Name:	PSU Access ID:

3. Say you have a flow network with integer capacities and are given an integer maximum flow f^*. Now say one of the edges of the network has its capacity increased by 1. Find an $O(|E|)$ algorithm that computes the maximum flow on this updated flow network.

   Solution: First note that $|V_f| = O(|E_f|)$, where V_f is the nodes in the given flow, and E_f is its edges since each node has at least 1 edge so the residual graph can be constructed in $O(|V_f|+|E_f|) = O(|E_f|)$ (ignoring nodes without at least one edge in V). Next an $s-t$ path can be found by running Explore from s in $O(|E_f|)$ and the path can be augmented in $O(|E_f|)$ if it is found, giving the new flow. Note that if no $s-t$ path is found, the max flow is the same, and that after augmenting, the max flow must increase by exactly one since we are working with integer capacities/flows and the FF algorithm (and that the flow can only increase by at most one when the capacity is increased by one), so this process only needs to happen once, giving a runtime of $O(|E|)$ since $E_f = O(|E|)$.

4. Let $G = (V, E)$ be a connected DAG, where each edge is annotated with some positive weight. Let s be a source vertex in G. Suppose we run Dijkstra's algorithm to compute the distance from s to each vertex $v \in V$, and then order the vertices in increasing order of their distance from s. Are we guaranteed that this is a valid topological sort of G (YES or NO)? Justify your answer.

   Solution: Consider the graph with 3 nodes - S, A, and B with edge weights w(S, A) = 2, w(A, B) = 3, and w(S, B) = 1. Valid topological order is S, A, and B but with the algorithm mentioned in the question it will be S, B, and A which is incorrect.

Name:	**PSU Access ID (xyz1234):**